Pemrograman Berorientasi Objek dengan C#

[Lecture Note Pertemuan ke - 15]
(LINQ / Lanjutan)

Ng Poi Wong, 2020, Sesi 7.b : LINQ (Lanjutan), Lecture Notes, Pemrograman Berorientasi Objek dengan C# (IF0058), STMIK Mikroskil Medan, Dikirimkan 02 Maret 2020.

Capaian MK : Mahasiswa mampu menerapkan LINQ

DAFTAR ISI PENJELASAN SLIDE

Slide 03 s/d 07	[Group]	2
Slide 08 & 09	[Take & Skip]	10
Slide 10 & 11	[Operator Set]	13
Slide 12 s/d 14	[Join]	17
Slide 15 s/d 29	[LINQ dengan Database]	21
Contoh - 1	[Contoh Sederhana LINQ to ADO.NET - 1]	22
Contoh - 2	[Contoh Sederhana LINQ to ADO.NET - 2]	24
Contoh - 3	[Contoh Sederhana LINQ to ADO.NET - 3]	26

PENJELASAN DARI SLIDE ke-03 s/d 07

- Klausa **group...by** merupakan operator dasar untuk mengelompokkan data berdasarkan field tertentu (kunci) sesuai keinginan user.

[Contoh Sederhana]

- Contoh sederhana dari penggunaan klausa **group...by** adalah menggunakan Console Application.
- Deklarasikan struktur berikut pada Console Application untuk **Contoh Sederhana ke-1** dan **Contoh Sederhana ke-2** :

```
struct Konsumen {
  public string Nama, Kota, Provinsi;
}
```

- Untuk **Contoh Sederhana ke-1**, diberikan kode program berikut :

```
List<Konsumen> pelanggan = new List<Konsumen>() {
  new Konsumen() {Nama = "Ivonne", Kota = "Medan",
    Provinsi = "Sumatera Utara"},
  new Konsumen() {Nama = "Bonni", Kota = "Banda Aceh", Provinsi = "Aceh"},
  new Konsumen() {Nama = "Gally", Kota = "Sibolga",
    Provinsi = "Sumatera Utara"},
  new Konsumen() {Nama = "Andi", Kota = "Bandung",
    Provinsi = "Jawa Barat"},
  new Konsumen() {Nama = "Kelly", Kota = "Meulaboh", Provinsi = "Aceh"},
  new Konsumen() {Nama = "Denada", Kota = "Medan",
    Provinsi = "Sumatera Utara"},
  new Konsumen() {Nama = "Halim", Kota = "Jakarta", Provinsi = "Jakarta"},
  new Konsumen() {Nama = "Celia", Kota = "Sibolga",
    Provinsi = "Sumatera Utara"},
  new Konsumen() {Nama = "Jovan", Kota = "Banda Aceh", Provinsi = "Aceh"},
  new Konsumen() {Nama = "Elfrida", Kota = "Medan",
    Provinsi = "Sumatera Utara"},
  new Konsumen() {Nama = "Monica", Kota = "Bandung",
    Provinsi = "Jawa Barat"},
  new Konsumen() {Nama = "Lumia", Kota = "Denpasar", Provinsi = "Bali"}
};
var hasil = from x in pelanggan
            group x by x.Kota into grpKota
            orderby grpKota.Count() descending, grpKota.Key
            select new {
              Kota = grpKota.Key,
              Jlh_Pelanggan = grpKota.Count()
            };
foreach (var i in hasil)
  Console.WriteLine("{0,-12}: {1} Pelanggan", i.Kota, i.Jlh_Pelanggan);
```

PENJELASAN DARI SLIDE ke-03 s/d 07

Kode program di atas memiliki penjelasan berikut :

1. Variabel bertipe List "**pelanggan**" dapat digambarkan menjadi tabel yang terdiri dari 3 (tiga) kolom (Nama, Kota, dan Provinsi) dan berisikan 12 data pelanggan. Gambaran tabel tersebut dapat diilustrasikan menjadi :

Nama	Kota	Provinsi
Ivonne	Medan	Sumatera Utara
Bonni	Banda Aceh	Aceh
Gally	Sibolga	Sumatera Utara
Andi	Bandung	Jawa Barat
Kelly	Meulaboh	Aceh
Denada	Medan	Sumatera Utara
Halim	Jakarta	Jakarta
Celia	Sibolga	Sumatera Utara
Jovan	Banda Aceh	Aceh
Elfrida	Medan	Sumatera Utara
Monica	Bandung	Jawa Barat
Lumia	Denpasar	Bali

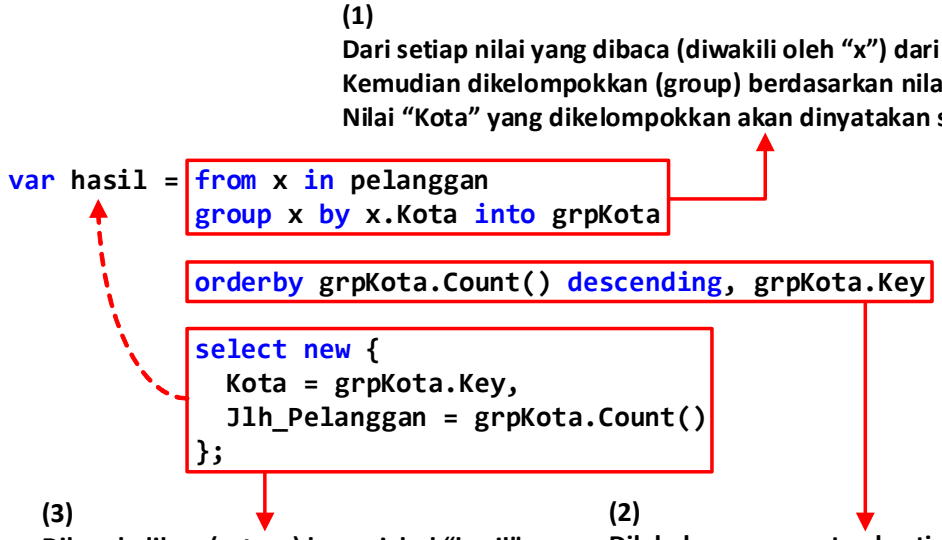
2. Dengan ilustrasi tabel dari isi variabel "**pelanggan**" di atas, contoh sederhana ini akan menyajikan informasi berupa **Jumlah Pelanggan berdasarkan Kota**, misalnya berapa jumlah pelanggan dari kota "Medan", berapa jumlah pelanggan dari kota "Banda Aceh", dan seterusnya.

Nama	Kota	Provinsi
Ivonne	Medan	Sumatera Utara
Denada	Medan	Sumatera Utara
Elfrida	Medan	Sumatera Utara
Gally	Sibolga	Sumatera Utara
Celia	Sibolga	Sumatera Utara
Bonni	Banda Aceh	Aceh
Jovan	Banda Aceh	Aceh
Kelly	Meulaboh	Aceh
Andi	Bandung	Jawa Barat
Monica	Bandung	Jawa Barat
Halim	Jakarta	Jakarta
Lumia	Denpasar	Bali

Untuk bisa mendapatkan informasi tersebut, maka data dari variabel "**pelanggan**" perlu dikelompokkan berdasarkan kolom "**Kota**".

PENJELASAN DARI SLIDE ke-03 s/d 07

3. Potongan kode program LINQ di atas memiliki penjelasan berikut :



(3)
Dikembalikan (return) ke variabel "hasil" berupa 2 (dua) nilai, yakni :
1. **Nilai "Kota" yang telah dikelompokkan, dimana dibaca dari method ".Key" dari "grpKota".**
2. **Jumlah Data (Jumlah Pelanggan) per "Kota" yang dibaca dari "grpKota".**

(2)
Dilakukan pengurutan bertingkat sebanyak 2 (dua) tingkat, dimana :
1. **"grpKota.Count() descending"**
   Tingkat pertama diurutkan secara Descending berdasarkan Jumlah Data (Jumlah Pelanggan) per "Kota" yang dibaca dari "grpKota".
2. **"grpKota.Key"**
   Tingkat ke-2 diurutkan secara Ascending berdasarkan nilai "Kota". Pada statement pengelompokan dari (1), nilai "x" dikelompokkan berdasarkan nilai "Kota" dan dinyatakan sebagai "grpKota", maka method ".Key" dari "grpKota" adalah sama dengan mengambil nilai "Kota" yang menjadi kunci pengelompokkan.

- Untuk **Contoh Sederhana ke-2**, diberikan kode program berikut :

```
List<Konsumen> pelanggan = new List<Konsumen>() {
  new Konsumen() {Nama = "Ivonne", Kota = "Medan",
    Provinsi = "Sumatera Utara"},
  new Konsumen() {Nama = "Bonni", Kota = "Banda Aceh", Provinsi = "Aceh"},
  new Konsumen() {Nama = "Gally", Kota = "Sibolga",
    Provinsi = "Sumatera Utara"},
  new Konsumen() {Nama = "Andi", Kota = "Bandung",
    Provinsi = "Jawa Barat"},
  new Konsumen() {Nama = "Kelly", Kota = "Meulaboh", Provinsi = "Aceh"},
  new Konsumen() {Nama = "Denada", Kota = "Medan",
    Provinsi = "Sumatera Utara"},
  new Konsumen() {Nama = "Halim", Kota = "Jakarta", Provinsi = "Jakarta"},
```

PENJELASAN DARI SLIDE ke-03 s/d 07

```
        new Konsumen() {Nama = "Celia", Kota = "Sibolga",
          Provinsi = "Sumatera Utara"},
        new Konsumen() {Nama = "Jovan", Kota = "Banda Aceh", Provinsi = "Aceh"},
        new Konsumen() {Nama = "Elfrida", Kota = "Medan",
          Provinsi = "Sumatera Utara"},
        new Konsumen() {Nama = "Monica", Kota = "Bandung",
          Provinsi = "Jawa Barat"},
        new Konsumen() {Nama = "Lumia", Kota = "Denpasar", Provinsi = "Bali"}
    };
    var hasil = from x in pelanggan
                group x by x.Provinsi into grpProv
                orderby grpProv.Count() descending, grpProv.Key
                select new {
                  Provinsi = grpProv.Key,
                  Jlh_Pelanggan = grpProv.Count()
                };
    foreach (var i in hasil)
      Console.WriteLine("{0,-15}: {1} Pelanggan", i.Provinsi, i.Jlh_Pelanggan);
```

Kode program di atas memiliki penjelasan berikut :

1. Variabel bertipe List "**pelanggan**" di atas memiliki gambaran ilustrasi tabel yang sama dengan **Contoh Sederhana ke-1**.
2. Dengan ilustrasi tabel dari isi variabel "**pelanggan**" dari **Contoh Sederhana ke-1**, contoh sederhana ini ingin menyajikan informasi berupa **Jumlah Pelanggan berdasarkan Provinsi**, misalnya berapa jumlah pelanggan dari provinsi "Sumatera Utara", berapa jumlah pelanggan dari provinsi "Aceh", dan seterusnya.

Nama	Kota	Provinsi
Bonni	Banda Aceh	Aceh
Kelly	Meulaboh	Aceh
Jovan	Banda Aceh	Aceh
Ivonne	Medan	Sumatera Utara
Gally	Sibolga	Sumatera Utara
Denada	Medan	Sumatera Utara
Celia	Sibolga	Sumatera Utara
Elfrida	Medan	Sumatera Utara
Andi	Bandung	Jawa Barat
Monica	Bandung	Jawa Barat
Halim	Jakarta	Jakarta
Lumia	Denpasar	Bali

Aceh : 3 pelanggan
Sumatera Utara : 5 pelanggan
Jawa Barat : 2 pelanggan
Jakarta : 1 pelanggan
Bali : 1 pelanggan

Untuk bisa mendapatkan informasi tersebut, maka data dari variabel "**pelanggan**" perlu dikelompokkan berdasarkan kolom "**Provinsi**".

PENJELASAN DARI SLIDE ke-03 s/d 07

3. Potongan kode program LINQ di atas memiliki penjelasan berikut :

(1)
Dari setiap nilai yang dibaca (diwakili oleh "x") dari variabel "pelanggan", Kemudian dikelompokkan (group) berdasarkan nilai "Provinsi" dari "x", Nilai "Provinsi" yang dikelompokkan akan dinyatakan sebagai "grpProv"

```
var hasil = from x in pelanggan
            group x by x.Provinsi into grpProv

            orderby grpProv.Count() descending, grpProv.Key

            select new {
              Provinsi = grpProv.Key,
              Jlh_Pelanggan = grpProv.Count()
            };
```

(3)
Dikembalikan (return) ke variabel "hasil" berupa 2 (dua) nilai, yakni :
1. Nilai "Provinsi" yang telah dikelompokkan, dimana dibaca dari method ".Key" dari "grpProv".
2. Jumlah Data (Jumlah Pelanggan) per "Provinsi" yang dibaca dari "grpProv".

(2)
Dilakukan pengurutan bertingkat sebanyak 2 (dua) tingkat, dimana :
1. "grpProv.Count() descending"
Tingkat pertama diurutkan secara Descending berdasarkan Jumlah Data (Jumlah Pelanggan) per "Provinsi" yang dibaca dari "grpProv".
2. "grpProv.Key"
Tingkat ke-2 diurutkan secara Ascending berdasarkan nilai "Provinsi". Pada statement pengelompokan dari (1), nilai "x" dikelompokkan berdasarkan nilai "Provinsi" dan dinyatakan sebagai "grpProv", maka method ".Key" dari "grpProv" adalah sama dengan mengambil nilai "Provinsi" yang menjadi kunci pengelompokkan.

- Untuk **Contoh Sederhana ke-3**, deklarasikan struktur berikut pada Console Application :

```
struct Produk {
  public int Kode;
  public string Nama, Jenis;
  public int Stok, HargaSatuan;
}
```

Kemudian tuliskan kode program berikut :

```
List<Produk> barang = new List<Produk>() {
  new Produk() {Kode = 65, Nama = "Gula", Jenis = "Sembako", Stok = 12,
    HargaSatuan = 12000},
  new Produk() {Kode = 12, Nama = "Kemeja", Jenis = "Pakaian", Stok = 5,
    HargaSatuan = 60000},
  new Produk() {Kode = 98, Nama = "Beras", Jenis = "Sembako", Stok = 30,
    HargaSatuan = 125000},
```

PENJELASAN DARI SLIDE ke-03 s/d 07

```
    new Produk() {Kode = 75, Nama = "Jaket", Jenis = "Pakaian", Stok = 6,
        HargaSatuan = 250000},
      new Produk() {Kode = 34, Nama = "Spidol", Jenis = "ATK", Stok = 75,
        HargaSatuan = 5000},
      new Produk() {Kode = 67, Nama = "Garam", Jenis = "Sembako", Stok = 30,
        HargaSatuan = 7500},
      new Produk() {Kode = 41, Nama = "Kertas", Jenis = "ATK", Stok = 20,
        HargaSatuan = 25000},
      new Produk() {Kode = 86, Nama = "Pulpen", Jenis = "ATK", Stok = 80,
        HargaSatuan = 3000},
      new Produk() {Kode = 73, Nama = "Jeans", Jenis = "Pakaian", Stok = 4,
        HargaSatuan = 175000},
      new Produk() {Kode = 92, Nama = "Pensil", Jenis = "ATK", Stok = 60,
        HargaSatuan = 2000},
      new Produk() {Kode = 56, Nama = "Tepung", Jenis = "Sembako", Stok = 18,
        HargaSatuan = 22500},
      new Produk() {Kode = 28, Nama = "Bawang", Jenis = "Sembako", Stok = 45,
        HargaSatuan = 45000}
};
var hasil = from x in barang
            group x by x.Jenis into Kategori
            orderby Kategori.Key
            select new {
              Jenis = Kategori.Key,
              Jlh_Barang = Kategori.Count(),
              Total_Stok = Kategori.Sum(x => x.Stok),
              Total_Modal = Kategori.Sum(x => x.Stok * x.HargaSatuan)
            };
foreach (var i in hasil)
  Console.WriteLine("{0,-8}: {1} Produk (Total Stok = {2},
    Total Modal = Rp. {3})",
    i.Jenis, i.Jlh_Barang, i.Total_Stok, i.Total_Modal);
```

Kode program di atas memiliki penjelasan berikut :

1. Variabel bertipe List "**barang**" dapat digambarkan menjadi tabel yang terdiri dari 5 (lima) kolom (Kode, Nama, Jenis, Stok, dan Harga Satuan) dan berisikan 12 data pelanggan. Gambaran tabel tersebut dapat diilustrasikan menjadi :

PENJELASAN DARI SLIDE ke-03 s/d 07

Kode	Nama	Jenis	Stok	Harga Satuan
65	Gula	Sembako	12	12000
12	Kemeja	Pakaian	5	60000
98	Beras	Sembako	30	125000
75	Jaket	Pakaian	6	250000
34	Spidol	ATK	75	5000
67	Garam	Sembako	30	7500
41	Kertas	ATK	20	25000
86	Pulpen	ATK	80	3000
73	Jeans	Pakaian	4	175000
92	Pensil	ATK	60	2000
56	Tepung	Sembako	18	22500
28	Bawang	Sembako	45	45000

2. Dengan ilustrasi tabel dari isi variabel "**barang**" di atas, contoh sederhana ini akan menyajikan informasi berupa **Jumlah Barang, Total Stok, dan Total Modal (Stok × Harga Satuan) berdasarkan Jenis**, misalnya untuk Jenis = "Pakaian" akan disajikan :

   Jumlah Barang = 3 produk, yakni Kode = 12, 75, dan 73
   Total Stok = 15 (5 Kemeja + 6 Jaket + 4 Jeans)
   Total Modal = Rp. 2500000 ((5 Kemeja × Rp. 60000) + (6 Jaket × Rp. 250000) + (4 Jeans × Rp. 175000))

Kode	Nama	Jenis	Stok	Harga Satuan
12	Kemeja	Pakaian	5	60000
75	Jaket	Pakaian	6	250000
73	Jeans	Pakaian	4	175000
34	Spidol	ATK	75	5000
41	Kertas	ATK	20	25000
86	Pulpen	ATK	80	3000
92	Pensil	ATK	60	2000
65	Gula	Sembako	12	12000
98	Beras	Sembako	30	125000
67	Garam	Sembako	30	7500
56	Tepung	Sembako	18	22500
28	Bawang	Sembako	45	45000

Jenis = "Pakaian"
Jumlah Barang = 3 produk
Total Stok = 5 + 6 + 4
Total Modal = (5 x 6000) + (6 x 250000) + (4 x 175000)

Jenis = "ATK"
Jumlah Barang = 4 produk
Total Stok = 75 + 20 + 80 + 60
Total Modal = (75 x 5000) + (20 x 25000) + (80 x 3000) + (60 x 2000)

Jenis = "Sembako"
Jumlah Barang = 5 produk
Total Stok = 12 + 30 + 30 + 18 + 45
Total Modal = (12 x 12000) + (30 x 125000) + (30 x 7500) + (18 x 22500) + (45 x 45000)

Untuk bisa mendapatkan informasi tersebut, maka data dari variabel "**barang**" perlu dikelompokkan berdasarkan kolom "**Jenis**".

PENJELASAN DARI SLIDE ke-03 s/d 07

3. Potongan kode program LINQ di atas memiliki penjelasan berikut :

(1)
Dari setiap nilai yang dibaca (diwakili oleh "x") dari variabel "barang", Kemudian dikelompokkan (group) berdasarkan nilai "Jenis" dari "x", Nilai "Jenis" yang dikelompokkan akan dinyatakan sebagai "Kategori", Kemudian diurutkan secara Ascending berdasarkan nilai "Jenis" sebagai kunci pengelompokkan yang diambil melalui method ".Key".

```
var hasil = from x in barang
            group x by x.Jenis into Kategori
            orderby Kategori.Key

            select new {
              Jenis = Kategori.Key,
              Jlh_Barang = Kategori.Count(),
              Total_Stok = Kategori.Sum(x => x.Stok),
              Total_Modal = Kategori.Sum(x => x.Stok * x.HargaSatuan)
            };
```

(2)
Dikembalikan (return) ke variabel "hasil" berupa 4 (empat) nilai, yakni :

1. **Nilai "Jenis" yang telah dikelompokkan, dimana dibaca dari method ".Key" dari "Kategori".**
2. **Jumlah Data (Jumlah Barang) per "Jenis" yang dibaca dari "Kategori".**
3. **Total Stok per "Jenis" yang dibaca dari "Kategori" dengan method "Sum(x => x.Stok)", dimana dibaca : Jumlahkan setiap nilai "Stok" yang diwakili oleh "x".**
4. **Total Modal per "Jenis" yang dibaca dari "Kategori" dengan method "Sum(x => x.Stok * x.HargaSatuan)", dimana dibaca : Jumlahkan hasil perkalian nilai "Stok" dengan nilai "HargaSatuan" yang diwakili oleh "x".**

PENJELASAN DARI SLIDE ke-08 & 09

- Operator **Take** merupakan method LINQ untuk mengambil sejumlah record teratas. Identik dengan SELECT TOP dari SQL.
- Operator **Skip** merupakan method LINQ untuk mengambil record dari kebalikan operator **Take**.
- Operator **Take** dan **Skip** hanya dapat digunakan pada Method Syntax.

[Contoh Sederhana]

- Contoh sederhana dari penggunaan operator **Take** dan **Skip** adalah menggunakan Console Application.
- Deklarasikan struktur berikut pada Console Application :

```
struct Produk {
  public int Kode;
  public string Nama, Jenis;
  public int Stok, HargaSatuan;
}
```

- Untuk **Contoh Sederhana** ini, diberikan kode program berikut :

```
List<Produk> barang = new List<Produk>() {
  new Produk() {Kode = 65, Nama = "Gula", Jenis = "Sembako", Stok = 12,
    HargaSatuan = 12000},
  new Produk() {Kode = 12, Nama = "Kemeja", Jenis = "Pakaian", Stok = 5,
    HargaSatuan = 60000},
  new Produk() {Kode = 98, Nama = "Beras", Jenis = "Sembako", Stok = 30,
    HargaSatuan = 125000},
  new Produk() {Kode = 75, Nama = "Jaket", Jenis = "Pakaian", Stok = 6,
    HargaSatuan = 250000},
  new Produk() {Kode = 34, Nama = "Spidol", Jenis = "ATK", Stok = 75,
    HargaSatuan = 5000},
  new Produk() {Kode = 67, Nama = "Garam", Jenis = "Sembako", Stok = 35,
    HargaSatuan = 7500},
  new Produk() {Kode = 41, Nama = "Kertas", Jenis = "ATK", Stok = 20,
    HargaSatuan = 25000},
  new Produk() {Kode = 86, Nama = "Pulpen", Jenis = "ATK", Stok = 80,
    HargaSatuan = 3000},
  new Produk() {Kode = 73, Nama = "Jeans", Jenis = "Pakaian", Stok = 4,
    HargaSatuan = 175000},
  new Produk() {Kode = 92, Nama = "Pensil", Jenis = "ATK", Stok = 60,
    HargaSatuan = 2000},
  new Produk() {Kode = 56, Nama = "Tepung", Jenis = "Sembako", Stok = 18,
    HargaSatuan = 22500},
  new Produk() {Kode = 28, Nama = "Bawang", Jenis = "Sembako", Stok = 45,
    HargaSatuan = 45000}
};
Console.WriteLine("3 Besar dengan Harga Satuan Termahal :");
foreach (Produk i in barang.OrderByDescending(x => x.HargaSatuan).Take(3))
  Console.WriteLine("{0,-3}{1,-7}({2,-7}) [Stok = {3:D2}, Harga = Rp.{4}]",
    i.Kode, i.Nama, i.Jenis, i.Stok, i.HargaSatuan);
Console.WriteLine("\nDaftar Stok (SELAIN 3 Besar Stok Terbanyak) :");
```

PENJELASAN DARI SLIDE ke-08 & 09

```
foreach (Produk i in barang.OrderByDescending(x => x.Stok).Skip(3))
  Console.WriteLine("{0,-3}{1,-7}({2,-7}) [Stok = {3:D2}, Harga = Rp.{4}]",
    i.Kode, i.Nama, i.Jenis, i.Stok, i.HargaSatuan);
```

Potongan kode program LINQ di atas memiliki penjelasan berikut :

1. Variabel bertipe List "**barang**" dapat digambarkan menjadi tabel yang terdiri dari 5 (lima) kolom (Kode, Nama, Jenis, Stok, dan Harga Satuan) dan berisikan 12 data pelanggan. Gambaran tabel tersebut dapat diilustrasikan menjadi :

Kode	Nama	Jenis	Stok	Harga Satuan
65	Gula	Sembako	12	12000
12	Kemeja	Pakaian	5	60000
98	Beras	Sembako	30	125000
75	Jaket	Pakaian	6	250000
34	Spidol	ATK	75	5000
67	Garam	Sembako	35	7500
41	Kertas	ATK	20	25000
86	Pulpen	ATK	80	3000
73	Jeans	Pakaian	4	175000
92	Pensil	ATK	60	2000
56	Tepung	Sembako	18	22500
28	Bawang	Sembako	45	45000

2. Untuk potongan kode program LINQ pertama berikut :

```
foreach (Produk i in barang.OrderByDescending(x => x.HargaSatuan).Take(3))
  Console.WriteLine("{0,-3}{1,-7}({2,-7}) [Stok = {3:D2}, Harga = Rp.{4}]",
    i.Kode, i.Nama, i.Jenis, i.Stok, i.HargaSatuan);
```

Potongan kode program LINQ pertama dalam bentuk Method Syntax ini dikombinasikan dengan statement perulangan foreach, dimana setiap nilai dari variabel "**barang**" (diwakili oleh "**x**") diurutkan secara Descending berdasarkan "**HargaSatuan**", kemudian hanya diambil 3 data teratas (3 besar dengan Harga Satuan termahal).

Kode	Nama	Jenis	Stok	Harga Satuan
75	Jaket	Pakaian	6	250000
73	Jeans	Pakaian	4	175000
98	Beras	Sembako	30	125000
12	Kemeja	Pakaian	5	60000
28	Bawang	Sembako	45	45000
41	Kertas	ATK	20	25000
56	Tepung	Sembako	18	22500
65	Gula	Sembako	12	12000
67	Garam	Sembako	35	7500
34	Spidol	ATK	75	5000
86	Pulpen	ATK	80	3000
92	Pensil	ATK	60	2000

Setelah diurutkan berdasarkan "Harga Satuan" secara Descending, Dikembalikan (return) 3 data teratas (3 besar dengan Harga Satuan termahal)

PENJELASAN DARI SLIDE ke-08 & 09

3. Untuk potongan kode program LINQ ke-2 berikut :

```
foreach (Produk i in barang.OrderByDescending(x => x.Stok).Skip(3))
   Console.WriteLine("{0,-3}{1,-7}({2,-7}) [Stok = {3:D2}, Harga = Rp.{4}]",
      i.Kode, i.Nama, i.Jenis, i.Stok, i.HargaSatuan);
```

Potongan kode program LINQ ke-2 dalam bentuk Method Syntax ini dikombinasikan dengan statement perulangan foreach, dimana setiap nilai dari variabel "**barang**" (diwakili oleh "**x**") diurutkan secara Descending berdasarkan "**Stok**", kemudian diambil semua data teratas, **KECUALI 3 (tiga) data teratas dengan Stok terbesar**.

Kode	Nama	Jenis	Stok	Harga Satuan
86	Pulpen	ATK	80	3000
34	Spidol	ATK	75	5000
92	Pensil	ATK	60	2000
28	Bawang	Sembako	45	45000
67	Garam	Sembako	35	7500
98	Beras	Sembako	30	125000
41	Kertas	ATK	20	25000
56	Tepung	Sembako	18	22500
65	Gula	Sembako	12	12000
75	Jaket	Pakaian	6	250000
12	Kemeja	Pakaian	5	60000
73	Jeans	Pakaian	4	175000

Setelah diurutkan berdasarkan "Stok" secara Descending, Dikembalikan (return) semua data teratas, KECUALI 3 (tiga) data teratas dengan Stok terbesar

PENJELASAN DARI SLIDE ke-10 & 11

- **Operator Set** pada LINQ terdiri dari 3 method, yakni **Union**, **Intersect**, dan **Except**.
- Method **Union** untuk mengambil hasil gabungan (∪) dari 2 sumber data, misalnya :
  A = { 1, 2, 3, 4, 5 }
  B = { 3, 5, 7, 9, 11 }
  A ∪ B = { 1, 2, 3, 4, 5, 7, 9, 11 }
  B ∪ A = { 1, 2, 3, 4, 5, 7, 9, 11 }
  Dimana A ∪ B = B ∪ A
- Method **Intersect** untuk mengambil hasil irisan (∩) dari 2 sumber data.
  A = { 1, 2, 3, 4, 5 }
  B = { 3, 5, 7, 9, 11 }
  A ∩ B = { 3, 5 }
  B ∩ A = { 3, 5 }
  Dimana A ∩ B = B ∩ A
- Method **Except** untuk mengambil hasil perbedaan (–) dari 2 sumber data.
  A = { 1, 2, 3, 4, 5 }
  B = { 3, 5, 7, 9, 11 }
  A – B = { 1, 2, 4 }
  B – A = { 7, 9, 11 }
  Dimana A – B ≠ B – A
- Koleksi data yang dibaca dengan **Operator Set**, wajib memiliki Tipe atau Struktur yang sama **(Union-Compatible)**.
- Apabila pada hasil dari **Operator Set**, terdapat nilai yang sama (duplikat), maka secara otomatis akan dihapuskan nilai yang sama tersebut, sehingga yang diambil adalah nilai yang unik.
- Method **Union**, **Intersect**, dan **Except** hanya dapat digunakan pada Method Syntax.

[Contoh Sederhana]

- Contoh sederhana dari penggunaan **Operator Set** adalah menggunakan Console Application.
- Untuk **Contoh Sederhana ke-1**, diberikan kode program berikut :

```
List<int> himpA = new List<int>() { 1, 2, 3, 4, 5 };
List<int> himpB = new List<int>() { 3, 5, 7, 9, 11 };
Console.WriteLine("A UNION B :");
foreach (int i in himpA.Union(himpB))
  Console.Write("{0} ", i);
Console.WriteLine("\n\nA INTERSECT B :");
foreach (int i in himpA.Intersect(himpB))
  Console.Write("{0} ", i);
Console.WriteLine("\n\nA EXCEPT B :");
foreach (int i in himpA.Except(himpB))
  Console.Write("{0} ", i);
Console.WriteLine("\n\nB EXCEPT A :");
foreach (int i in himpB.Except(himpA))
  Console.Write("{0} ", i);
```

PENJELASAN DARI SLIDE ke-10 & 11

Potongan kode program LINQ di atas memiliki penjelasan berikut :

1. Untuk potongan LINQ pertama :

```
foreach (int i in himpA.Union(himpB))
  Console.Write("{0} ", i);
```

Potongan kode program LINQ pertama dalam bentuk Method Syntax ini dikombinasikan dengan statement perulangan foreach, dimana digunakan operator **Union** untuk menggabungkan semua nilai dari variabel "**himpA**" dan "**himpB**" **(himpA ∪ himpB)**.

2. Untuk potongan LINQ ke-2 :

```
foreach (int i in himpA.Intersect(himpB))
  Console.Write("{0} ", i);
```

Potongan kode program LINQ ke-2 dalam bentuk Method Syntax ini dikombinasikan dengan statement perulangan foreach, dimana digunakan operator **Intersect** untuk mengambil nilai irisan antara nilai dari variabel "**himpA**" dan "**himpB**" **(himpA ∩ himpB)**.

3. Untuk potongan LINQ ke-3 & 4 :

```
foreach (int i in himpA.Except(himpB))
  Console.Write("{0} ", i);
```

dan

```
foreach (int i in himpB.Except(himpA))
  Console.Write("{0} ", i);
```

Potongan kode program LINQ ke-3 dan ke-4 dalam bentuk Method Syntax ini dikombinasikan dengan statement perulangan foreach, dimana digunakan operator **Except** untuk mengambil nilai yang ada pada variabel "**himpA**", tetapi tidak ada pada variabel "**himpB**" **(himpA – himpB)**, kemudian mengambil nilai yang ada pada variabel "**himpB**", tetapi tidak ada pada variabel "**himpA**" **(himpB – himpA)**.

- Untuk **Contoh Sederhana ke-2**, deklarasikan struktur berikut pada Console Application :

```
struct Produk {
  public int Kode;
  public string Nama, Jenis;
}
```

Kemudian tuliskan kode program berikut :

```
List<Produk> listBrg1 = new List<Produk>() {
  new Produk() {Kode = 34, Nama = "Spidol", Jenis = "ATK"},
  new Produk() {Kode = 41, Nama = "Kertas", Jenis = "ATK"},
  new Produk() {Kode = 86, Nama = "Pulpen", Jenis = "ATK"},
  new Produk() {Kode = 92, Nama = "Pensil", Jenis = "ATK"},
  new Produk() {Kode = 65, Nama = "Gula", Jenis = "Sembako"},
  new Produk() {Kode = 73, Nama = "Jeans", Jenis = "Pakaian"}
};
```

PENJELASAN DARI SLIDE ke-10 & 11

```
List<Produk> listBrg2 = new List<Produk>() {
   new Produk() {Kode = 65, Nama = "Gula", Jenis = "Sembako"},
   new Produk() {Kode = 98, Nama = "Beras", Jenis = "Sembako"},
   new Produk() {Kode = 67, Nama = "Garam", Jenis = "Sembako"},
   new Produk() {Kode = 12, Nama = "Kemeja", Jenis = "Pakaian"},
   new Produk() {Kode = 75, Nama = "Jaket", Jenis = "Pakaian"},
   new Produk() {Kode = 73, Nama = "Jeans", Jenis = "Pakaian"}
};
Console.WriteLine("listBrg1 UNION listBrg2 :");
foreach (Produk i in listBrg1.Union(listBrg2).OrderBy(x => x.Jenis))
   Console.WriteLine("{0,-3:D}{1,-8}{2}", i.Kode, i.Nama, i.Jenis);
Console.WriteLine("\nlistBrg1 INTERSECT listBrg2 :");
foreach (Produk i in listBrg1.Intersect(listBrg2).OrderBy(x => x.Jenis))
   Console.WriteLine("{0,-3:D}{1,-8}{2}", i.Kode, i.Nama, i.Jenis);
Console.WriteLine("\nlistBrg1 EXCEPT listBrg2 :");
foreach (Produk i in listBrg1.Except(listBrg2).OrderBy(x => x.Jenis))
   Console.WriteLine("{0,-3:D}{1,-8}{2}", i.Kode, i.Nama, i.Jenis);
Console.WriteLine("\nlistBrg2 EXCEPT listBrg1 :");
foreach (Produk i in listBrg2.Except(listBrg1).OrderBy(x => x.Jenis))
   Console.WriteLine("{0,-3:D}{1,-8}{2}", i.Kode, i.Nama, i.Jenis);
```

Potongan kode program LINQ di atas memiliki penjelasan berikut :

1. Untuk potongan LINQ pertama :

   ```
   foreach (Produk i in listBrg1.Union(listBrg2).OrderBy(x => x.Jenis))
     Console.WriteLine("{0,-3:D}{1,-8}{2}", i.Kode, i.Nama, i.Jenis);
   ```

   Potongan kode program LINQ pertama dalam bentuk Method Syntax ini dikombinasikan dengan statement perulangan foreach, dimana digunakan operator **Union** untuk menggabungkan semua data dari variabel "**listBrg1**" dan "**listBrg2**", kemudian diurutkan secara Ascending berdasarkan "**Jenis**" dari setiap nilai yang diwakili oleh "**x**".
2. Untuk potongan LINQ ke-2 :

   ```
   foreach (Produk i in listBrg1.Intersect(listBrg2).OrderBy(x => x.Jenis))
     Console.WriteLine("{0,-3:D}{1,-8}{2}", i.Kode, i.Nama, i.Jenis);
   ```

   Potongan kode program LINQ pertama dalam bentuk Method Syntax ini dikombinasikan dengan statement perulangan foreach, dimana digunakan operator **Intersect** untuk mengambil nilai irisan antara data dari variabel "**listBrg1**" dan "**listBrg2**", kemudian diurutkan secara Ascending berdasarkan "**Jenis**" dari setiap nilai yang diwakili oleh "**x**".
3. Untuk potongan LINQ ke-3 & 4 :

   ```
   foreach (Produk i in listBrg1.Except(listBrg2).OrderBy(x => x.Jenis))
     Console.WriteLine("{0,-3:D}{1,-8}{2}", i.Kode, i.Nama, i.Jenis);
   ```

   dan

   ```
   foreach (Produk i in listBrg2.Except(listBrg1).OrderBy(x => x.Jenis))
     Console.WriteLine("{0,-3:D}{1,-8}{2}", i.Kode, i.Nama, i.Jenis);
   ```

PENJELASAN DARI SLIDE ke-10 & 11

Potongan kode program LINQ ke-3 dan ke-4 dalam bentuk Method Syntax ini dikombinasikan dengan statement perulangan foreach, dimana digunakan operator **Except** untuk mengambil data yang ada pada variabel "**listBrg1**", tetapi tidak ada pada variabel "**listBrg2**", kemudian mengambil nilai yang ada pada variabel "**listBrg2**", tetapi tidak ada pada variabel "**listBrg1**".
Dari ke-2 hasil operator Except tersebut, kemudian diurutkan secara Ascending berdasarkan "**Jenis**" dari setiap nilai yang diwakili oleh "**x**".

PENJELASAN DARI SLIDE ke-12 s/d 14

- Klausa **join** merupakan operator dasar untuk menggabungkan 2 sumber data. Identik dengan INNER JOIN pada SQL yang menggabungkan 2 tabel atau lebih.
- Perbedaan klausa **join** dengan **Operator Set [Slide 10 & 11]** adalah :
  1. Klausa **join** tidak berlaku aturan **Union-Compatible**.
  2. Klausa **join** wajib memerlukan 1 field sebagai penghubung antara 2 sumber data (dalam Basis Data, diilustrasikan sebagai Relationship-Key, berupa Primary-Key dan Foreign-Key).

[Contoh Sederhana]

- Contoh sederhana dari penggunaan klausa **join** adalah menggunakan Console Application.
- Deklarasikan 2 (dua) struktur berikut pada Console Application :

```
struct Produk {
  public int Kode;
  public string Nama, Jenis;
}
struct Stock {
  public int Kode, Stok, Harga;
}
```

- Untuk **Contoh Sederhana** ini, diberikan kode program berikut :

```
List<Produk> barang = new List<Produk>(){
  new Produk() {Kode = 65, Nama = "Gula", Jenis = "Sembako"},
  new Produk() {Kode = 12, Nama = "Kemeja", Jenis = "Pakaian"},
  new Produk() {Kode = 98, Nama = "Beras", Jenis = "Sembako"},
  new Produk() {Kode = 41, Nama = "Kertas", Jenis = "ATK"},
  new Produk() {Kode = 86, Nama = "Pulpen", Jenis = "ATK"},
  new Produk() {Kode = 92, Nama = "Pensil", Jenis = "ATK"}
};
List<Stock> gudang = new List<Stock>(){
  new Stock() {Kode = 65, Stok = 12, Harga = 18000},
  new Stock() {Kode = 86, Stok = 75, Harga = 4500},
  new Stock() {Kode = 65, Stok = 18, Harga = 12000},
  new Stock() {Kode = 86, Stok = 85, Harga = 3000},
  new Stock() {Kode = 12, Stok = 7, Harga = 80000},
  new Stock() {Kode = 98, Stok = 15, Harga = 135000},
  new Stock() {Kode = 41, Stok = 24, Harga = 40000},
  new Stock() {Kode = 98, Stok = 20, Harga = 110000},
  new Stock() {Kode = 41, Stok = 35, Harga = 30000},
  new Stock() {Kode = 86, Stok = 70, Harga = 6000}
};
```

PENJELASAN DARI SLIDE ke-12 s/d 14

```
var hasil = from i in barang
            join j in gudang
            on i.Kode equals j.Kode
            orderby i.Kode
            select new {
              Kode = i.Kode,
              Nama = i.Nama,
              Jenis = i.Jenis,
              Stok = j.Stok,
              HargaSatuan = j.Harga
            };
foreach (var i in hasil)
  Console.WriteLine("{0,-3}{1,-7}({2,-7}) [Stok = {3:D2}, Harga = Rp.{4}]",
    i.Kode, i.Nama, i.Jenis, i.Stok, i.HargaSatuan);
```

Potongan kode program LINQ di atas memiliki penjelasan berikut :

1. Variabel bertipe List "**barang**" dapat digambarkan menjadi tabel yang terdiri dari 3 (tiga) kolom (Kode, Nama, dan Jenis) dan berisikan 6 data barang. Gambaran tabel tersebut dapat diilustrasikan menjadi :

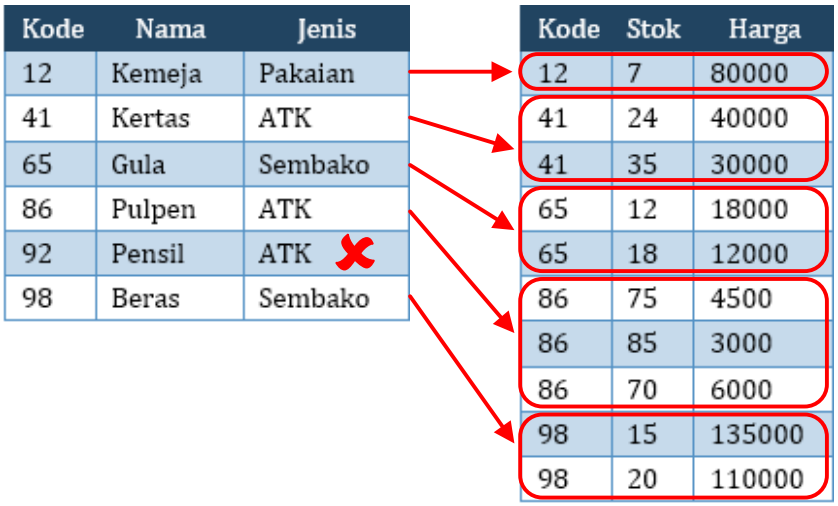
Kode	Nama	Jenis
65	Gula	Sembako
12	Kemeja	Pakaian
98	Beras	Sembako
41	Kertas	ATK
86	Pulpen	ATK
92	Pensil	ATK

2. Variabel bertipe List "**gudang**" dapat digambarkan menjadi tabel yang terdiri dari 3 (tiga) kolom (Kode, Stok, dan Harga) dan berisikan 10 data stok di dalam gudang. Gambaran tabel tersebut dapat diilustrasikan menjadi :

Kode	Stok	Harga
65	12	18000
86	75	4500
65	18	12000
86	85	3000
12	7	80000
98	15	135000
41	24	40000
98	20	110000
41	35	30000
86	70	6000

PENJELASAN DARI SLIDE ke-12 s/d 14

3. Dengan ilustrasi 2 (dua) tabel dari isi variabel "**barang**" dan "**gudang**" di atas, contoh sederhana ini akan menyajikan informasi berupa menggabungkan ke-2 tabel tersebut menjadi 1 (satu) tabel yang terdiri dari 5 (lima) kolom (Kode, Nama, Jenis, Stok, dan Harga).



Kode	Nama	Jenis
12	Kemeja	Pakaian
41	Kertas	ATK
65	Gula	Sembako
86	Pulpen	ATK
92	Pensil	ATK
98	Beras	Sembako

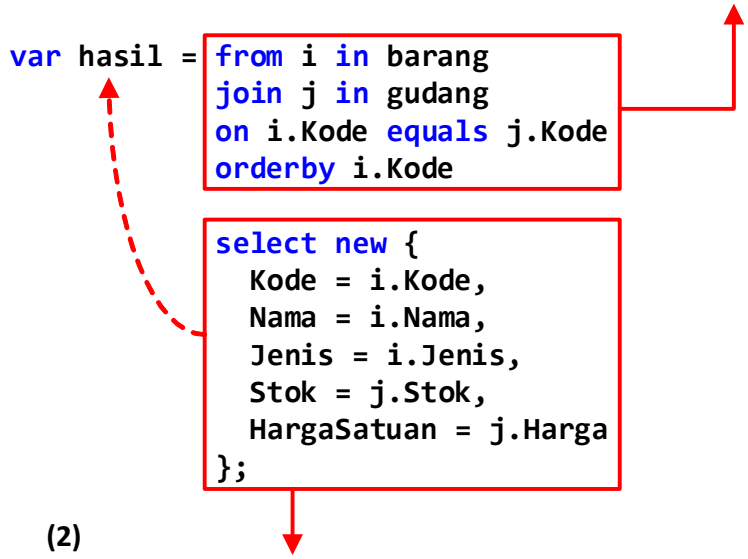
Kode	Stok	Harga
12	7	80000
41	24	40000
41	35	30000
65	12	18000
65	18	12000
86	75	4500
86	85	3000
86	70	6000
98	15	135000
98	20	110000

Penggabungan 2 (dua) tabel menggunakan klausa **join** dengan field/kolom penghubung antara 2 (dua) tabel tersebut adalah "**Kode**".

PENJELASAN DARI SLIDE ke-12 s/d 14

4. Potongan kode program LINQ di atas memiliki penjelasan berikut :

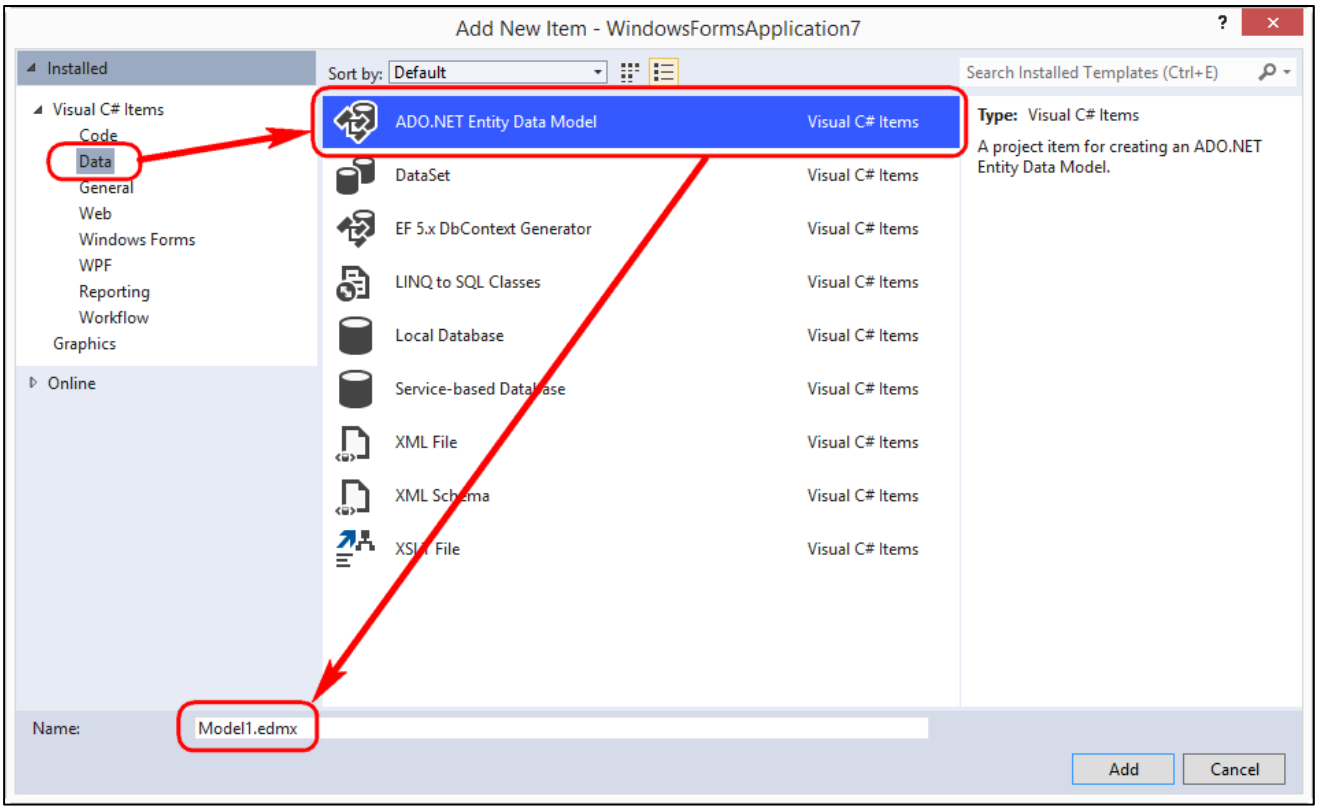
(1)
Setiap nilai yang dibaca (diwakili oleh "i") dari variabel "barang",
di-JOIN-kan dengan setiap nilai yang dibaca (diwakili oleh "j") dari variabel "gudang",
dengan field/kolom penghubung antara "i" dan "j" adalah "Kode".
Hasil JOIN kemudian diurutkan berdasarkan "Kode" secara Ascending.



(2)
Dikembalikan (return) ke variabel "hasil" berupa 5 (lima) nilai, yakni :
1. Nilai "Kode" yang dibaca dari "Kode" yang diwakili oleh "i" (variabel "barang").
2. Nilai "Nama" yang dibaca dari "Nama" yang diwakili oleh "i" (variabel "barang").
3. Nilai "Jenis" yang dibaca dari "Jenis" yang diwakili oleh "i" (variabel "barang").
4. Nilai "Stok" yang dibaca dari "Stok" yang diwakili oleh "j" (variabel "gudang").
5. Nilai "HargaSatuan" yang dibaca dari "Harga" yang diwakili oleh "j" (variabel "gudang").

PENJELASAN DARI SLIDE ke-15 s/d 29

- Untuk dapat menggunakan **LINQ to ADO.NET**, terlebih dahulu harus dibentuk **Model Entitas Data (Entity Data Model)** dari database dan tabel yang akan dilibatkan di dalam **LINQ to ADO.NET**.
- Untuk membuat **Entity Data Model**, dapat dilakukan melalui :
  1. Pilih menu **Project | Add New Item**, atau klik kanan nama Projects pada jendela Solution Explorer, pilih menu **Add | New Item**
  2. Akan muncul jendela **Add New Item**, kemudian pada sisi kiri, pilih sub menu **Data**, dan pada sisi tengah tampilan akan ada pilihan **ADO.NET Entity Data Model**.
  3. Sesuaikan nama Entity Data Model yang diinginkan, kemudian klik **Add**.



- Untuk langkah-langkah konfigurasi dari pembuatan **Model Entitas Data (Entity Data Model)** akan dijelaskan melalui Video YouTube berikut : https://youtu.be/ChDA-66ywi0

CONTOH SEDERHANA LINQ to ADO.NET - 1

Untuk contoh kasus sederhana dari penggunaan **LINQ to ADO.NET (1)**, dapat dilihat contoh berikut :

1. Buatkan 1 (satu) Console Application.

2. Tambahkan 1 (satu) **Entity Data Model** dari database "**Northwind**" dengan nama "**dbNorthwind**" sesuai dengan Video YouTube di atas, dengan nama alias Connection String adalah "**NorthwindEntities**".

3. Tabel yang terlibat ke dalam **Entity Data Model** "**dbNorthwind**" adalah tabel "**Customers**", "**Orders**", dan "**Order Details**".

4. Kemudian kode program berikut ke dalam Main Program :

```
NorthwindEntities nwe = new NorthwindEntities();
var query = from c in nwe.Customers
            join o in nwe.Orders
            on c.CustomerID equals o.CustomerID
            join od in nwe.Order_Details
            on o.OrderID equals od.OrderID
            select new {
              CustomerID = c.CustomerID,
              ContactName = c.ContactName,
              OrderID = o.OrderID,
              Qty = od.Quantity,
              UnitPrice = od.UnitPrice
            };
foreach (var i in query)
  Console.WriteLine("{0} {1} ({2} / {3} / {4})",
    i.CustomerID, i.ContactName, i.OrderID, i.Qty, i.UnitPrice);
```

CONTOH SEDERHANA LINQ to ADO.NET - 1

5. Kode program di atas memiliki penjelasan berikut :

(1)
Setiap record yang dibaca (diwakili "c") dari tabel "Customers", di-JOIN-kan dengan setiap record yang dibaca (diwakili "o") dari tabel "Orders", dengan field penghubung antara "c" dan "o" adalah field "CustomerID".

```
var query = from c in nwe.Customers
            join o in nwe.Orders
            on c.CustomerID equals o.CustomerID

            join od in nwe.Order_Details
            on o.OrderID equals od.OrderID

            select new {
              CustomerID = c.CustomerID,
              ContactName = c.ContactName,
              OrderID = o.OrderID,
              Qty = od.Quantity,
              UnitPrice = od.UnitPrice
            };
```

(2)
Dari hasil JOIN (1), di-JOIN-kan lagi dengan setiap record yang dibaca Setiap record yang dibaca (diwakili "od") dari tabel "Order Details", dengan field penghubung antara "o" dan "od" adalah field "OrderID".

(2)
Dikembalikan (return) ke variabel "hasil" berupa 5 (lima) nilai, yakni :
1. "CustomerID" yang dibaca dari field "CustomerID" dari tabel "Customers" (diwakili "c").
2. "ContactName" yang dibaca dari field "ContactName" dari tabel "Customers" (diwakili "c").
3. "OrderID" yang dibaca dari field "OrderID" dari tabel "Orders" (diwakili "o").
4. "Qty" yang dibaca dari field "Quantity" dari tabel "Order Details" (diwakili "od").
5. "UnitPrice" yang dibaca dari field "UnitPrice" dari tabel "Order Details" (diwakili "od").

CONTOH SEDERHANA LINQ to ADO.NET - 2

Untuk contoh kasus sederhana dari penggunaan **LINQ to ADO.NET (2)**, dapat dilihat contoh berikut :

1. Buatkan 1 (satu) Console Application.

2. Tambahkan 1 (satu) **Entity Data Model** dari database "**Northwind**" dengan nama "**dbNorthwind**" sesuai dengan Video YouTube di atas, dengan nama alias Connection String adalah "**NorthwindEntities**".

3. Tabel yang terlibat ke dalam **Entity Data Model** "**dbNorthwind**" adalah tabel "**Products**" dan "**Suppliers**".

4. Kemudian kode program berikut ke dalam Main Program :

```
NorthwindEntities nwe = new NorthwindEntities();
var query = from s in nwe.Suppliers
            select new {
              SupplierID = s.SupplierID,
              CompanyName = s.CompanyName,
              Country = s.Country,
              Products = s.Products
            };
foreach (var i in query)
  Console.WriteLine("{0:D2} {1,-38} ({2,-11}) [{3} Products Sold]",
    i.SupplierID, i.CompanyName, i.Country, i.Products.Count());
```

CONTOH SEDERHANA LINQ to ADO.NET - 2

5. Kode program di atas memiliki penjelasan berikut :

Baca setiap record (diwakili "s") dari tabel "Suppliers",
Kemudian dikembalikan ke variabel "hasil" berupa 4 (empat) nilai, yakni :
1. "SupplierID" yang dibaca dari field "SupplierID".
2. "CompanyName" yang dibaca dari field "CompanyName".
3. "Country" yang dibaca dari field "Country".
4. "Products" berisikan tabel "Products" yang merupakan hasil dari navigasi antar tabel yang memiliki relasi dengan tabel "Suppliers".

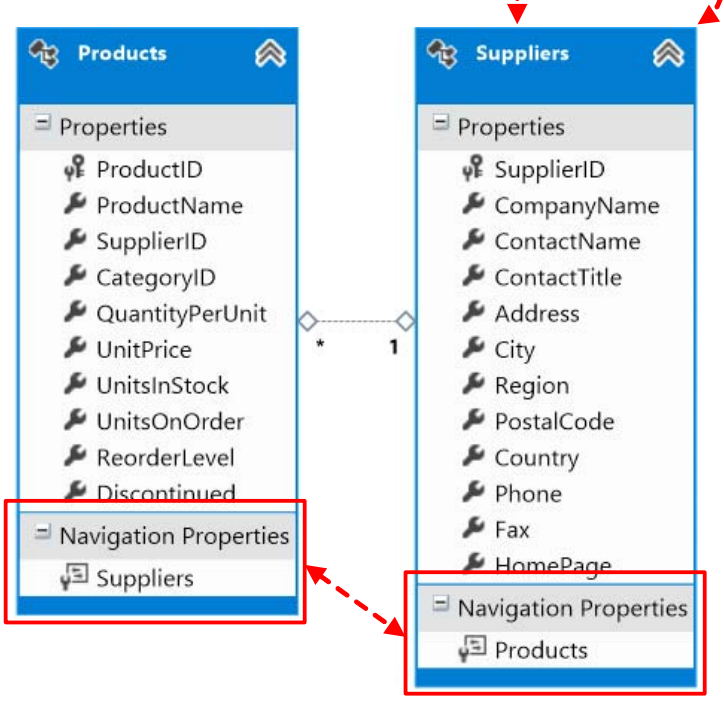
```
var query = from s in nwe.Suppliers
            select new {
              SupplierID = s.SupplierID,
              CompanyName = s.CompanyName,
              Country = s.Country,

              Products = s.Products

            };

foreach (var i in query)
  Console.WriteLine("{0:D2} {1,-38} ({2,-11}) [{3} Products Sold]",
    i.SupplierID, i.CompanyName, i.Country, i.Products.Count());
```

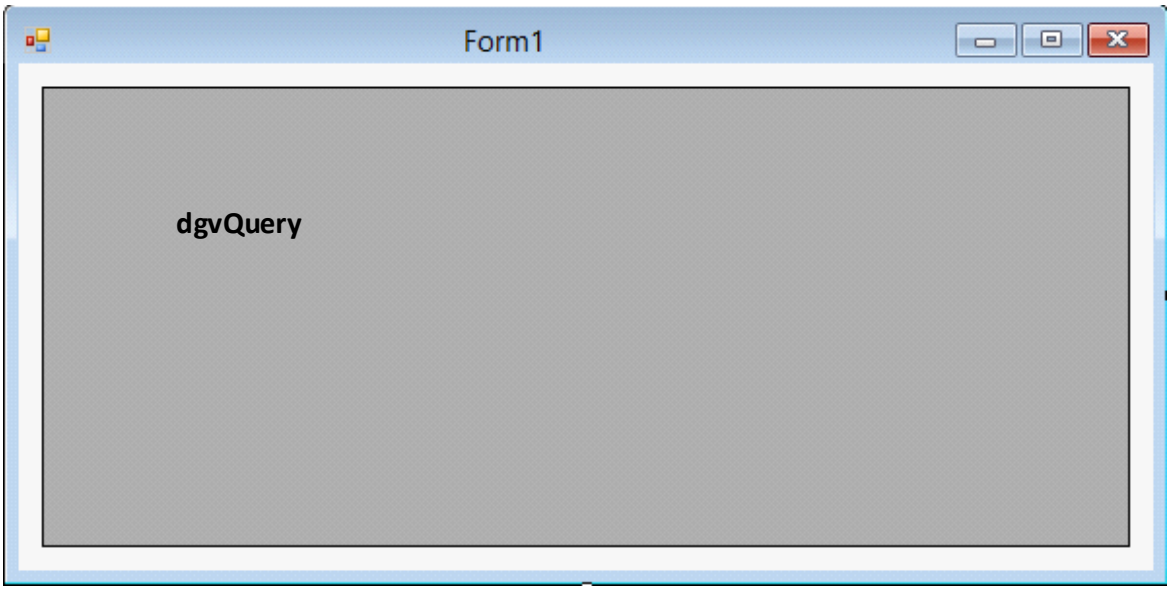
Salah satu keunggulan dari LINQ to ADO.NET adalah dapat dilakukan navigasi antar tabel yang saling berelasi. Dalam contoh ini, dikarenakan tabel "Products" dan "Suppliers" saling berelasi, maka jika dipanggil tabel "Products", maka dapat diakses juga isi tabel "Suppliers", dan sebaliknya, tanpa perlu digunakan JOIN.



CONTOH SEDERHANA LINQ to ADO.NET - 3

Untuk contoh kasus sederhana dari penggunaan **LINQ to ADO.NET (3)**, dapat dilihat contoh berikut :

1. Rancang tampilan Windows Form Application sebagai berikut :



2. Untuk masing-masing objek dari rancangan tampilan di atas, memiliki keterangan **Properties** sebagai berikut :
   a. Untuk **Properties** dari **Form** :

Properties	Keterangan
Font	Name = Consolas Size = 12 Style = Bold

   b. Untuk **Properties** dari **DataGrivView** "**dgvQuery**" :

Properties	Keterangan
Name	"dgvQuery"

3. Tambahkan 1 (satu) **Entity Data Model** dari database "**Northwind**" dengan nama "**dbNorthwind**" sesuai dengan Video YouTube di atas, dengan nama alias Connection String adalah "**NorthwindEntities**".

4. Tabel yang terlibat ke dalam **Entity Data Model** "**dbNorthwind**" adalah tabel "**Customers**", "**Orders**", "**Order Details**", dan "**Product**".

5. Tampilan program memiliki deklarasi variable global yakni :

```
NorthwindEntities nwe = new NorthwindEntities();
```

Dideklarasikan objek Connection String "**nwe**" dari Entity Data Model "**dbNorthwind**".

CONTOH SEDERHANA LINQ to ADO.NET - 3

6. Untuk rancangan tampilan di atas, memiliki **Event** "**Load**" dari **Form** dengan kode program sebagai berikut :

```
var query = from c in nwe.Customers
            join o in nwe.Orders
            on c.CustomerID equals o.CustomerID
            join od in nwe.Order_Details
            on o.OrderID equals od.OrderID
            join p in nwe.Products
            on od.ProductID equals p.ProductID
            orderby c.ContactName
            select new {
              CustomerID = c.CustomerID,
              ContactName = c.ContactName,
              OrderID = o.OrderID,
              OrderDate = o.OrderDate,
              ProductName = p.ProductName,
              Qty = od.Quantity,
              UnitPrice = od.UnitPrice
            };
this.Text = String.Format("Total {0} Record", query.Count());
dgvQuery.AllowUserToAddRows = false;
dgvQuery.AllowUserToDeleteRows = false;
dgvQuery.ReadOnly = true;
dgvQuery.DataSource = query.ToList();
```

Penjelasan untuk kode program di atas adalah :

- Untuk potongan kode program LINQ :

```
var query = from c in nwe.Customers
            join o in nwe.Orders
            on c.CustomerID equals o.CustomerID
            join od in nwe.Order_Details
            on o.OrderID equals od.OrderID
            join p in nwe.Products
            on od.ProductID equals p.ProductID
            orderby c.ContactName
            select new {
              CustomerID = c.CustomerID,
              ContactName = c.ContactName,
              OrderID = o.OrderID,
              OrderDate = o.OrderDate,
              ProductName = p.ProductName,
              Qty = od.Quantity,
              UnitPrice = od.UnitPrice
            };
```

CONTOH SEDERHANA LINQ to ADO.NET - 3

Setiap record yang dibaca (diwakili "**c**") dari tabel "**Customers**" di-JOIN-kan dengan setiap record yang dibaca (diwakili "**o**") dari tabel "**Orders**", dengan field penghubung antara "**c**" dan "**o**" adalah field "**CustomerID**".

Dari hasil JOIN 2 (dua) tabel di atas ("**c**" dan "**o**"), di-JOIN-kan lagi dengan setiap record yang dibaca (diwakili "**od**") dari tabel "**Order Details**", dengan field penghubung antara hasil JOIN ("**c**" dan "**o**") dan "**od**" adalah field "**OrderID**".

Dari hasil JOIN 3 (tiga) tabel di atas ("**c**", "**o**", dan "**od**"), di-JOIN-kan lagi dengan setiap record yang dibaca (diwakili "**p**") dari tabel "**Products**", dengan field penghubung antara hasil JOIN ("**c**", "**o**", dan "**od**") dan "**p**" adalah field "**ProductID**".

Dari hasil JOIN 4 (empat) tabel di atas ("**c**", "**o**", "**od**", dan "**p**"), kemudian diurutkan secara Ascending berdasarkan field "**ContactName**" dari tabel "**Customers**" (diwakili "**c**").

Dari hasil pengurutan terhadap hasil JOIN 4 (empat) tabel ("**c**", "**o**", "**od**", dan "**p**"), kemudian dikembalikan (return) ke variabel "**query**" berupa 7 (tujuh) nilai, yakni :

1. "**CustomerID**" yang dibaca dari field "**CustomerID**" dari tabel "**Customers**" (diwakili "**c**").
2. "**ContactName**" yang dibaca dari field "**ContactName**" dari tabel "**Customers**" (diwakili "**c**").
3. "**OrderID**" yang dibaca dari field "**OrderID**" dari tabel "**Orders**" (diwakili "**o**").
4. "**OrderDate**" yang dibaca dari field "**OrderDate**" dari tabel "**Orders**" (diwakili "**o**").
5. "**ProductName**" yang dibaca dari field "**ProductName**" dari tabel "**Products**" (diwakili "**p**").
6. "**Qty**" yang dibaca dari field "**Quantity**" dari tabel "**Order Details**" (diwakili "**od**").
7. "**UnitPrice**" yang dibaca dari field "**UnitPrice**" dari tabel "**Order Details**" (diwakili "**od**").

- Untuk potongan kode program :

```
this.Text = String.Format("Total {0} Record", query.Count());
```

Dicetak ke properties Text dari Form berupa informasi jumlah record hasil query LINQ yang ada pada variabel "**query**".

- Untuk potongan kode program :

```
dgvQuery.AllowUserToAddRows = false;
dgvQuery.AllowUserToDeleteRows = false;
dgvQuery.ReadOnly = true;
dgvQuery.DataSource = query.ToList();
```

Mengnon-aktifkan fitur menambahkan baris baru pada **DataGridView** "**dgvQuery**".

Mengnon-aktifkan fitur menghapus baris baru pada **DataGridView** "**dgvQuery**".

Mengaktifkan fitur Read-Only pada **DataGridView** "**dgvQuery**".

Menyatakan sumber data dari **DataGridView** "**dgvQuery**" adalah daftar record hasil query LINQ yang ada pada variabel "**query**".